TEMA 3: SISTEMAS ARITMÉTICOS

4. SUMADORES EN BINARIO PURO (I)

OPERACIONES EN BINARIO PURO

Suma en BINARIO PURO

- La tabla de la suma de dos bits (dígitos) binarios a y b, es la siguiente:

b/a	0	1
0	0	1
1	1	10

- La combinación a=1, b=1 produce un resultado de 10, es decir:

  El bit suma es 0 y existe acarreo ("me llevo" 1)

4. SUMADORES EN BINARIO PURO (II)

- Aplicando repetidamente esta tabla se pueden sumar dos números cualesquiera A y B. Para ello, se sumarán los bits de igual posición, teniendo en cuenta el acarreo de la etapa de bit anterior.

  Ejemplo:

$$\begin{array}{rrrrrrrr} & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ & 0 & 1 & 0 & 1 & 0 & 1 & 0 \; + \\ \hline 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}$$

  - Nota: el resultado puede tener un bit más que los sumandos. (desbordamiento)

4. SUMADORES EN BINARIO PURO (III)

Resta en BINARIO PURO

- La tabla de la resta de dos bits binarios a y b, a-b, es la siguiente:

b/a	0	1
0	0	1
1	11	0

- El 1, indica acarreo negativo.

- La combinación a=0, b=1 produce desbordamiento (a<b), por lo que aparece un acarreo negativo ("me llevo" -1) y el el resultado es 1.

  El bit suma es 1 y existe acarreo ("me llevo" 1)

-

4. SUMADORES EN BINARIO PURO (IV)

- Para restar dos cantidades **A-B**, primero se comprueba que **A≥B**, (igual que en la base decimal); si esta condición no se cumple, hay que realizar la operación inversa **–(B-A)**.

  Ejemplo:

$$\begin{array}{r} 1\ 1\ 0\ 0\ 1\ 0\ 1 \\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ - \\ \hline 0\ 0\ 0\ 1\ 0\ 1\ 1 \end{array}$$

  Nota: si se garantiza que **A≥B**, la resta **A-B** ocupará un número de bits igual o menor que **A** (no hay desbordamiento)

4.1 SEMISUMADOR DE UN BIT (SS) (HA) (I)

El semisumador (SS) de un bit o Half Adder (HA), es un circuito capaz de sumar dos bits aislados que proceden de dos operandos.

Entendemos por sumar "bits aislados" aquella operación que no tiene en cuenta los acarreos producidos en etapas anteriores.

Tabla de verdad para la suma de dos operandos *a* y *b* de un solo bit:

a_i, b_i: bits i-ésimos de los operandos de entrada.
s_i: resultado de la suma.
c_i: acarreo de la suma.

a_i	b_i	s_i	c_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

4.1 SEMISUMADOR DE UN BIT (SS) (HA) (II)

Estructura del SEMISUMADOR (S.S)

La función suma es:

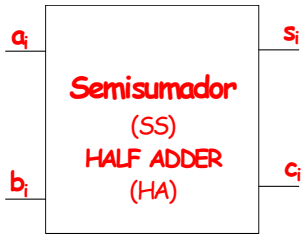
$s = \overline{a}b + a\overline{b} = a \oplus b$ XOR(a,b)

La función acarreo es:

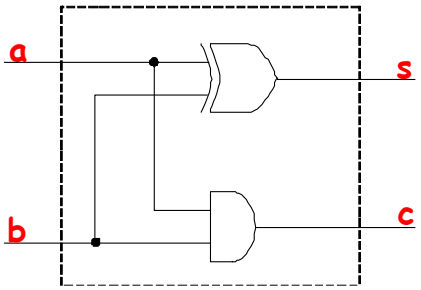
$c = ab$ AND (A,B)

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Diagrama de bloques



Implementación con P.L.B.



4.1 SUMADOR COMPLETO (SC) (FA) (I)

El sumador completo (SS) o Full Adder (FA), es un circuito que suma dos bits que proceden de dos operandos.

En este caso se tiene en cuenta el acarreo proveniente de etapas anteriores.

Tabla de verdad:

a_i, b_i: bits i-ésimos de los operandos de entrada.

s_i: bit i del resultado de la suma.

c_i: acarreo i de la suma.

a_i	b_i	c_{i-1}	s_i	c_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4.1 SUMADOR COMPLETO (SC) (FA) (II)

ECUACIONES

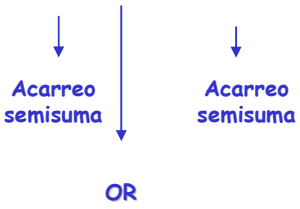
La función suma es:

$s_i = a_i \oplus b_i \oplus c_{i-1}$

La función acarreo es:

$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$

$= a_i b_i + c_{i-1} (a_i + b_i)$



a_i	b_i	c_{i-1}	s_i	c_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

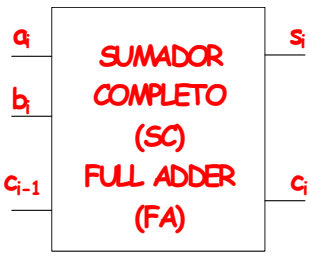
4.1 SUMADOR COMPLETO (SC) (FA) (III)

Estructura del SUMADOR COMPLETO (SC)

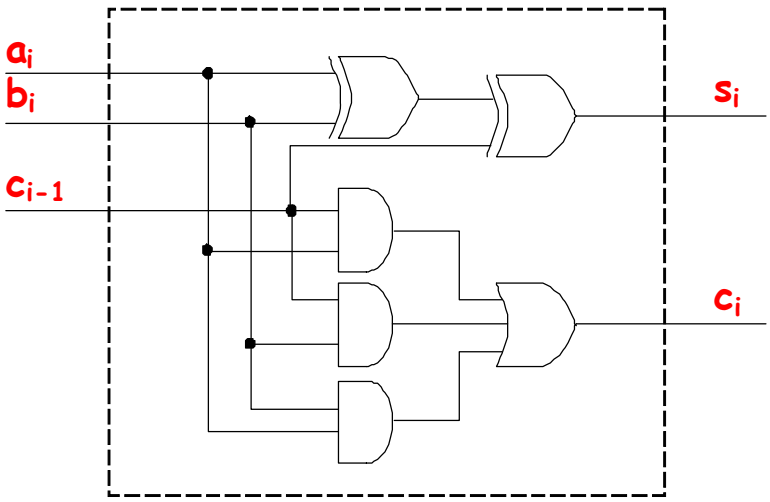
$s_i = a_i \oplus b_i \oplus c_{i-1}$

$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$

Diagrama de bloques



Implementación con puertas lógicas



4.1 SUMADOR COMPLETO (SC) (FA) (IV)

Estructura del SUMADOR COMPLETO (SC)

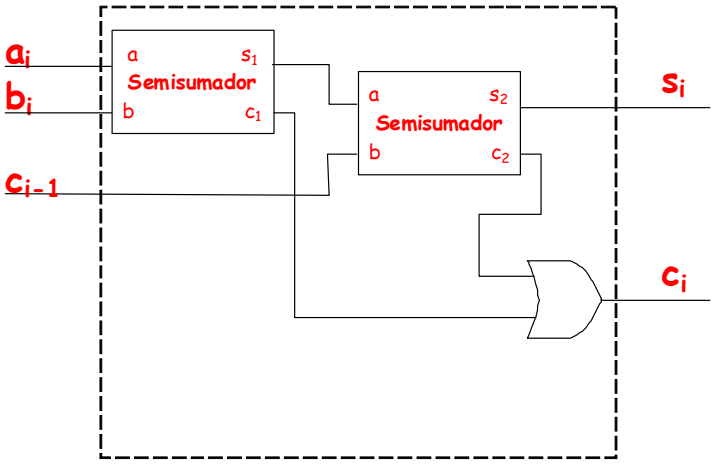
Implementación con semisumadores y puertas lógicas

La función acarreo es:

$$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$$

$$= a_i b_i + c_{i-1} (a_i + b_i)$$

Acarreo semisuma — Acarreo semisuma — OR



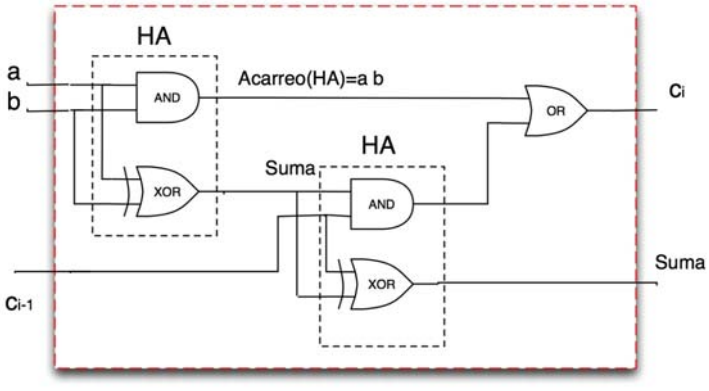
4.1 SUMADOR COMPLETO (SC) (FA) (V)

Estructura del SUMADOR COMPLETO (SC)

La función acarreo es:

$$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$$

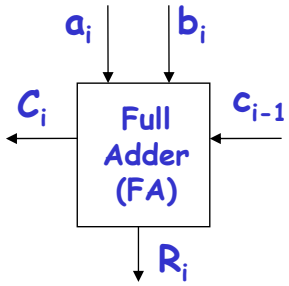
$$= a_i b_i + c_{i-1} (a_i + b_i)$$



Implementación de los SS con puertas lógicas

4.1 DESCRIPCIÓN VHDL DEL SUMADOR COMPLETO (VI)

Descripción de un sumador completo (full Adder (HA)) en estilo de Flujo de Datos



$$R_i = a_i \otimes b_i \otimes c_{i-1}$$

$$c_i = a_i b_i + c_{i-1}(a_i \otimes b_i)$$

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fulladd IS
        PORT (c(i-1), a(i), b(i): IN STD_LOGIC ;
              R(i), C(i): OUT STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
   BEGIN
        R(i) <= a(i) XOR b(i) XOR c(i-1) ;
        C(i) <= (a(i) AND b(i)) OR (c(i-1) AND a(i))
                OR (c(i-1) AND b(i)) ;
END LogicFunc ;
```

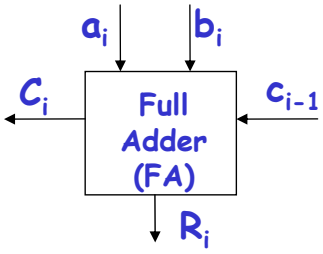

4.1 DESCRIPCIÓN VHDL DEL SUMADOR COMPLETO (VII)

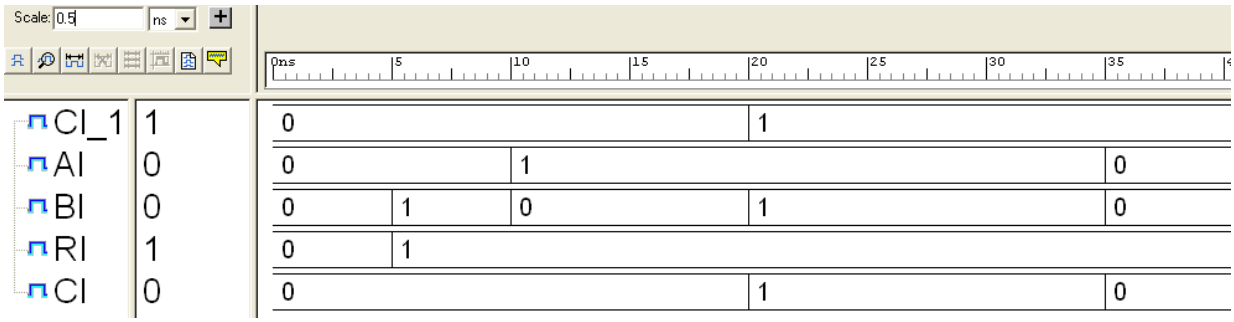
Arquitectura de test

```
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.all;

ENTITY simula_test IS
END simula_test;

ARCHITECTURE test_sumadorpp_arq OF simula_test IS
   COMPONENT fulladd
        PORT (c(i-1), a(i), b(i): IN STD_LOGIC ; R(i), C(i): OUT STD_LOGIC ) ;
   END COMPONENT;

   SIGNAL c(i-1), a(i), b(i),R(i), C(i): std_logic ;

   BEGIN
        I0: fulladd  PORT MAP (c(i-1), a(i), b(i), R(i), C(i));

        c(i-1) <= '0', '0' AFTER 10 ns, '1' AFTER 20 ns, '0' AFTER 30 ns;
        a(i) <= '0', '1' AFTER 5 ns,  '0' AFTER 15 ns, '1' AFTER 25 ns;
        b(i) <= '0', '1' AFTER 15 ns,  '0' AFTER 20 ns, '1' AFTER 25 ns;
END test_sumadorpp_arq;
```

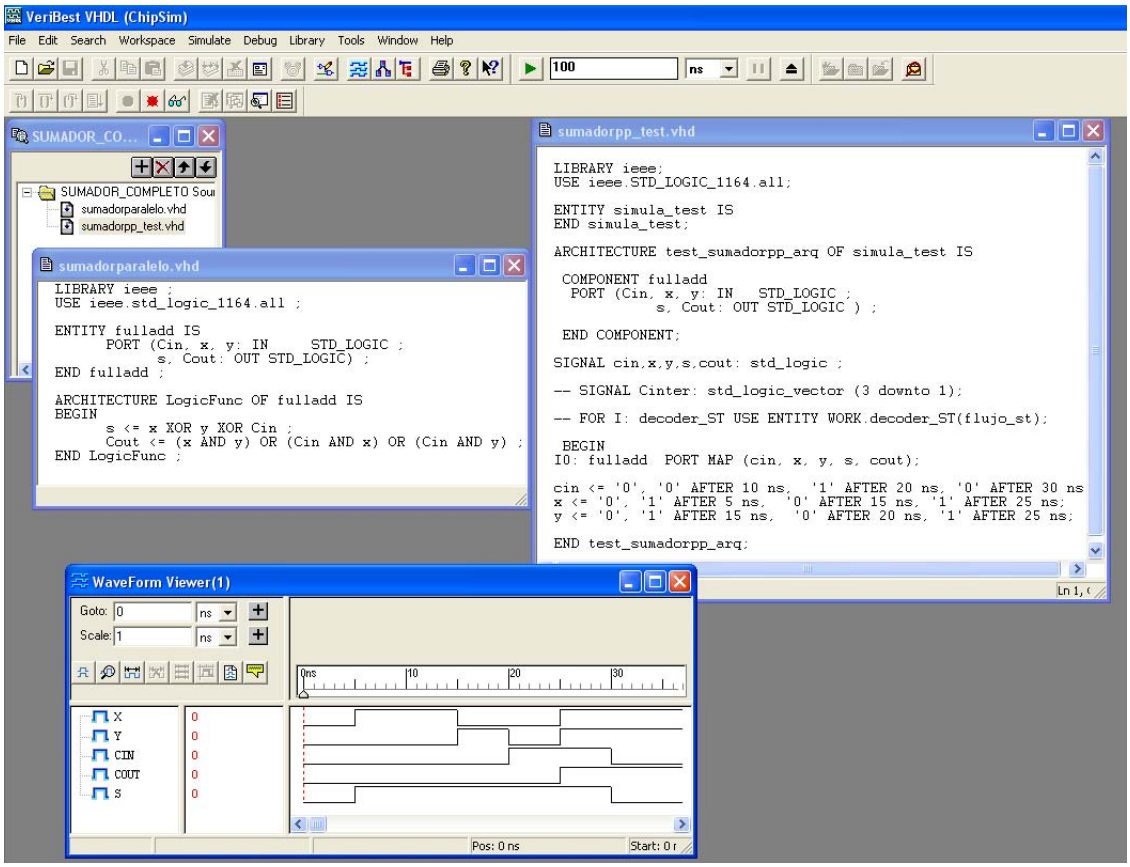
4.1 DESCRIPCIÓN VHDL DEL SUMADOR COMPLETO (VII)

Resultados de la Simulación de un sumador completo en estilo de Flujo de Datos

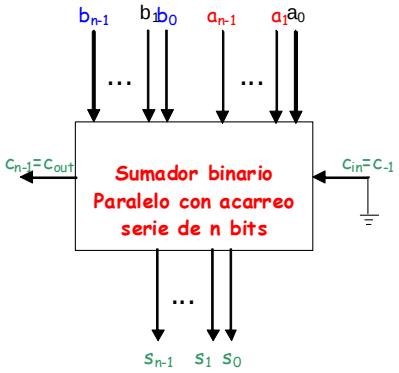




4.1 DESCRIPCIÓN VHDL DEL SUMADOR COMPLETO (IX)



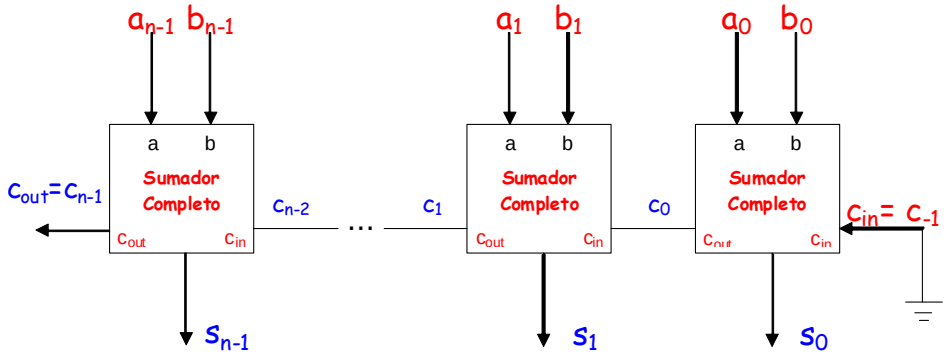
4.2 SUMADOR BINARIO PARALELO CON ACARREO EN SERIE DE n bits (I)



Tiene: n entradas de bit de datos a_i
n salidas de bit de datos b_i
acarreo de entrada c_{-1}
acarreo de salida c_{n-1}

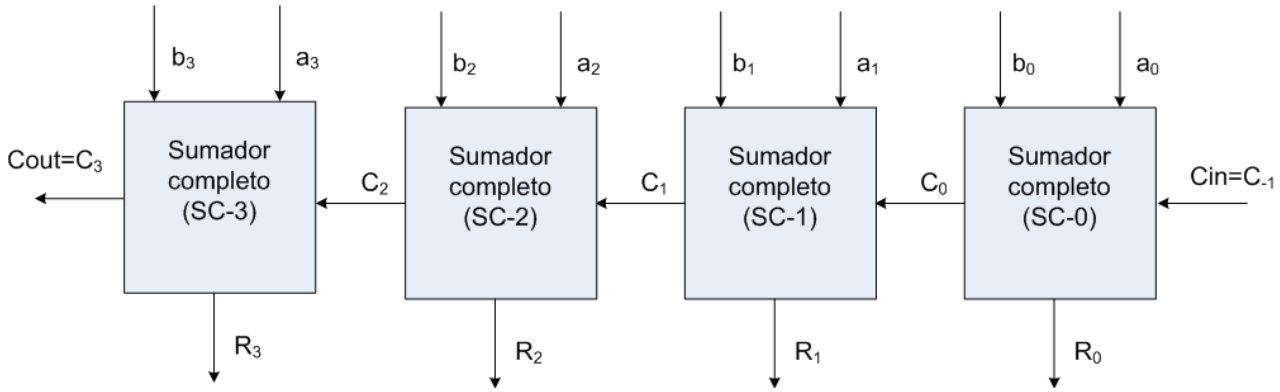
Problema: El retardo total del sumador es la suma de los retardos de los sumadores completos de un bit, debido a la transmisión serie de los acarreos.

Estructura basada en sumadores completos

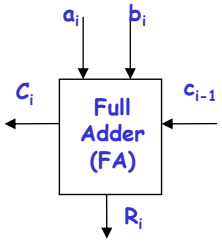


4.2. SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (II)

Estructura de 4 bits:



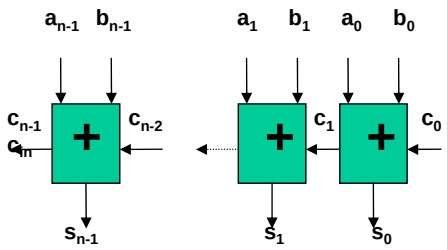
Basado en el sumador completo:



4.2. DESCRIPCIÓN VHDL DEL SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (III)

Descripción en estilo Estructural para 4 bits

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all;
ENTITY adder4 IS
   PORT ( cin: IN STD_LOGIC ;
          a, b: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          s : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
          cout : OUT STD_LOGIC );
END adder4;
ARCHITECTURE structural_4 OF adder4 IS
    COMPONENT fulladd IS  PORT (Cin, a, b: IN STD_LOGIC ; s, Cout: OUT STD_LOGIC ) ;
    END COMPONENT;
--Señal auxiliar que permite conectar los acarreos
    SIGNAL  C : STD_LOGIC_VECTOR(1 TO 3) ;
BEGIN
-- Instanciación de componentes
etapa0: fulladd PORT MAP (cin => cin,  a=>a(0),b=> b(0),s=>s(0),cout => c(1) ) ;
etapa1: fulladd PORT MAP (cin => c(1), a=>a(1),b=> b(1),s=>s(1),cout => c(2) ) ;
etapa2: fulladd PORT MAP (cin => c(2), a =>a(2),b=>b(2),s=>s(2),cout => c(3) ) ;
etapa3: fulladd PORT MAP (cin => c(3), a=>a(3), b=>b(3),s=> s(3),cout => Cout ) ;
END structural_4;
```



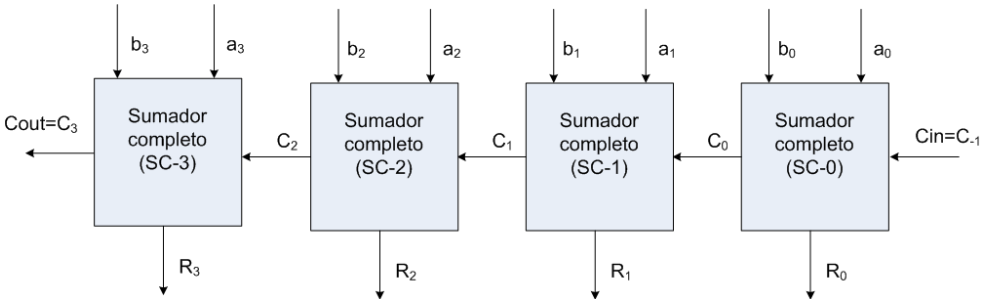
4.2. DESCRIPCIÓN VHDL DEL SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (IV)

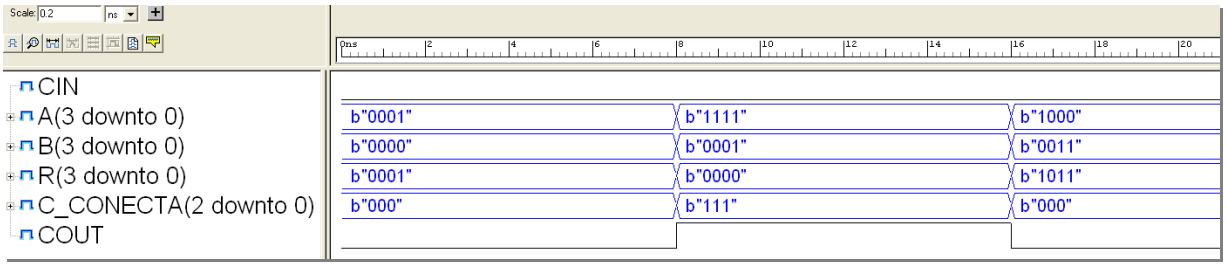
ARQUITECTURA DE TEST DEL SUMADOR BINARIO DE 4 BITS

```
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.all;

ENTITY simula_test IS
END simula_test;

ARCHITECTURE test_sumadorpp_arq OF simula_test IS
  COMPONENT adder4
    PORT ( cin : IN STD_LOGIC ;
                 a, b : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                 s: OUT STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                cout: OUT STD_LOGIC ) ;
  END COMPONENT;

  SIGNAL cin,cout: std_logic ;
  SIGNAL a,b,s:std_logic_vector(3 downto 0);
    BEGIN
      I0: adder4  PORT MAP (cin, a, b, s, cout);
      cin <= '0', '0' AFTER 10 ns,  '1' AFTER 20 ns, '0' AFTER 30 ns;
      a <= "0000", "0101" AFTER 5 ns,"1010" AFTER 15 ns, "1000" AFTER 25 ns;
      b <= "0000", "1001" AFTER 15 ns,  "0001" AFTER 20 ns, "0100" AFTER 25 ns;
END test_sumadorpp_arq;
```

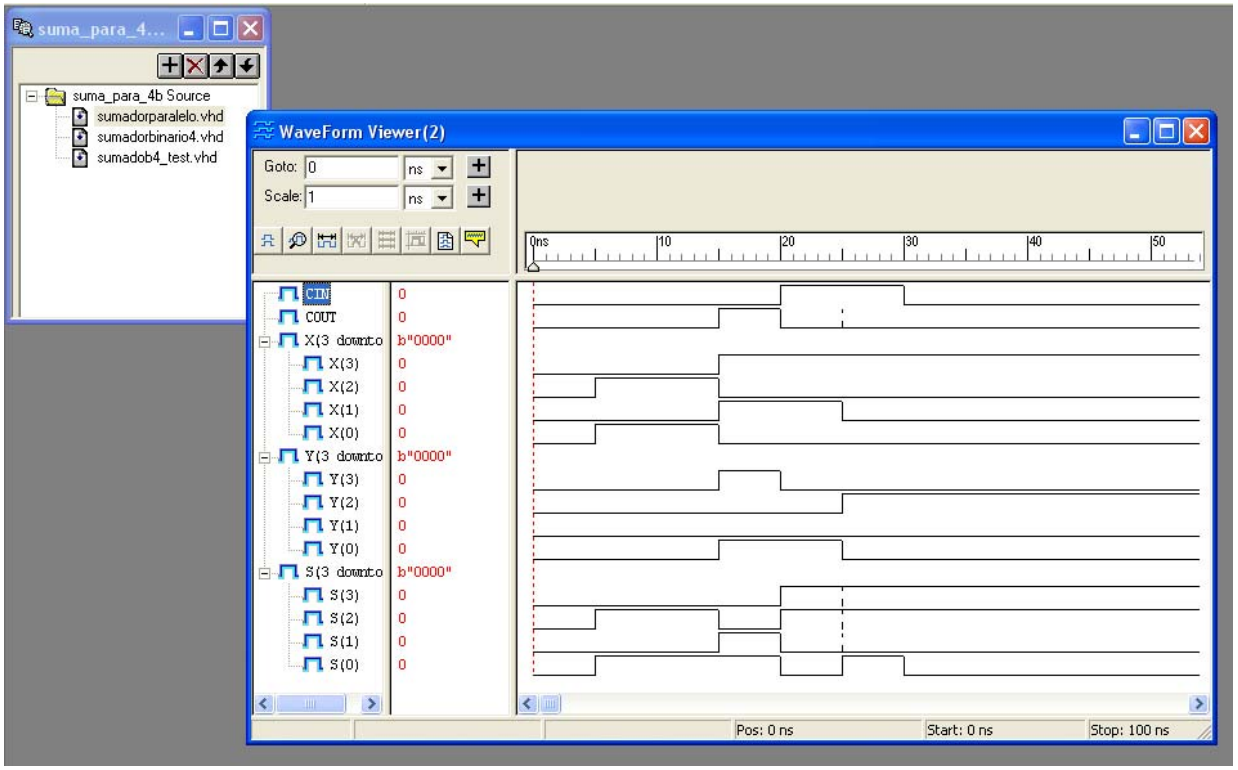
4.2 DESCRIPCIÓN VHDL DEL SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (V)

Resultados de la Simulación de un sumador binario paralelo con acarreo en serie de 4 bits





4.2 DESCRIPCIÓN VHDL DEL SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (VI)



SENTENCIA GENERATE en VHDL

- Equivale a un bucle hardware.
- Se usa para replicar partes del modelo.
- Es útil para describir series de componentes: sumadores binarios paralelo registros, memorias,…….

- **SINTAXIS**

1) etiqueta: **IF** condición **GENERATE**
   [declaraciones]
   BEGIN
   [declaraciones]
   END GENERATE [etiqueta];

2) etiqueta: **FOR** parametroRepetitivo **GENERATE**
   [declaraciones]
   BEGIN
   [declaraciones]
   END GENERATE [etiqueta];

SENTENCIA GENERATE en VHDL

EJEMPLO 1

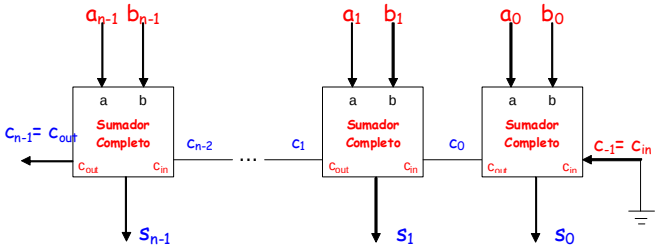
```
G0: IF n<4 GENERATE
        U1: and2 PORT MAP (e1,e0,s)
    END GENERATE;
```

EJEMPLO 2

```
ARCHITECTURE arq0 OF registroParaleloParaleloCuatroBits IS
    COMPONENT biestableD
        PORT(d, clk, clear: IN BIT; q: OUT BIT);
    END COMPONENT;
    FOR ALL: biestableD USE ENTITY WORK.biestableD(bD_flanco_b)
BEGIN
    G1: FOR i IN 0 TO 3 GENERATE
        Ui: biestableD PORT MAP(entDatos(i),clk,clear,salDatos(i));
    END GENERATE;
END registroParaleloParaleloCuatroBits;
```

4.2. DESCRIPCIÓN VHDL DEL SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (VI)

La entidad de un sumador binario con un número de bits genérico, podríamos realizarla introduciendo GENERIC:



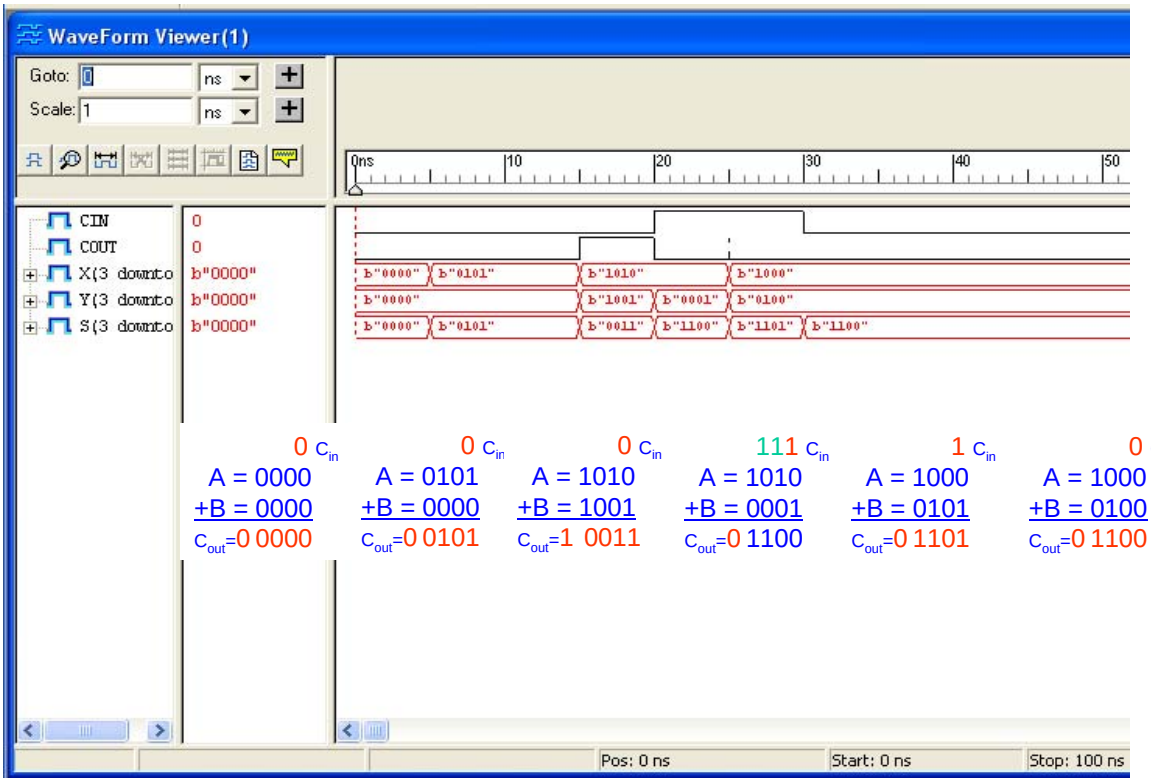

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY adder_gen IS
    GENERIC (num_bits: NATURAL);
    PORT ( cin : IN STD_LOGIC ;
           a, b : IN STD_LOGIC_VECTOR(num_bits-1 DOWNTO 0) ;
           -- (para 4 bits) a, b : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           s  : OUT  STD_LOGIC_VECTOR(num_bits-1 DOWNTO 0) ;
           -- (para 4 bits) s  : OUT  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           cout : OUT STD_LOGIC ) ;
END adder_gen ;
```


4.2. DESCRIPCIÓN VHDL DEL SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (VIII)

La arquitectura del sumador binario para un número de bits genérico (n bits), podríamos realizarla introduciendo GENERATE:

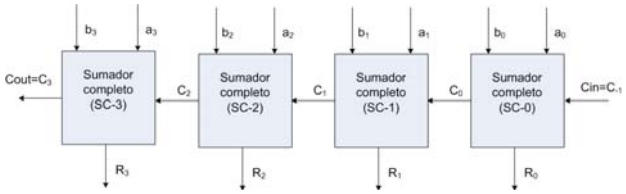
```
ARCHITECTURE structural OF adder_gen IS
      COMPONENT fulladd IS
         PORT (Cin, ai, bi: IN STD_LOGIC ; Ri, Ci : OUT STD_LOGIC);
      END COMPONENT;
--Señal auxiliar para conectar los acarreos intermedios
SIGNAL C_conecta: STD_LOGIC_VECTOR (0 TO num_bits -2);
   BEGIN
--Generación de Sumadores completos
       Repite: FOR I IN 0 TO num_bits -1 GENERATE
--Instanciación y conexión del primer SC-0
          primero_SC: IF (I = 0) GENERATE
           P: fulladd PORT MAP (Cin => Cin, ai => a(i), bi => b(i), Ri=> R(i), Ci =>C_conecta (i));
          END GENERATE;
--Instanciación y conexión de los SC intermedios del 1 al n-2
          medios_SC: IF (i /= 0 AND i/= num_bits -1) GENERATE
          M: fulladd PORT MAP (Cin => C_conecta (i-1), ai => a(i), bi => b(i), Ri=> R(i), Ci =>C_conecta (i));
          END GENERATE;
--Instanciación y conexión del SC final n-2
          final_SC: IF (i = num_bits -1) GENERATE
           F: fulladd PORT MAP (Cin => C_conecta (i-1), ai => a(i), bi => b(i), Ri=> R(i), Ci =>Cout);
          END GENERATE;
      END GENERATE;
END structural;
```

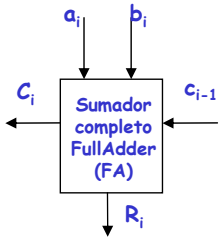
4.2. DESCRIPCIÓN VHDL DEL SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (IX)



0 C_{in}	0 C_{in}	0 C_{in}	111 C_{in}	1 C_{in}	0 C_{in}
A = 0000	A = 0101	A = 1010	A = 1010	A = 1000	A = 1000
+B = 0000	+B = 0000	+B = 1001	+B = 0001	+B = 0101	+B = 0100
C_{out}=0 0000	C_{out}=0 0101	C_{out}=1 0011	C_{out}=0 1100	C_{out}=0 1101	C_{out}=0 1100

4.2. SUMADOR BINARIO PARALELO CON ACARREO EN SERIE de n bits (X)





CÁLCULO DE RETARDOS

Suponemos que cada celda básica de un SC tiene un retardo T_{ci}= 1 ns en la obtención del acarreo de salida c_i y un retardo T_{ri}=2 ns en la obtención del resultado R_i:

Retardos c_i	Retardos R_i
T_{co} = 1 ns	T_{r0} = 2 ns
T_{c1} = 1 ns+T_{co}= 2 ns	T_{r1} = 2 ns+ T_{co} = 3 ns
T_{c2} = 1 ns+T_{c1}= 3 ns	T_{r2} = 2 ns+ T_{c1} = 4 ns
T_{c3} = 1 ns+T_{c2}= 4 ns	T_{r3} = 2 ns+ T_{c2}= 5 ns

Cuanto mayor es la dimensión del sumador mayor es el retardo

Para n bits: acarreo C_{out}= C_n tiene un retardo T_{cn-1}= n T_{c0}

resultado R_n tiene un retardo T_{RESUL} =T_{rn} = T_{r0} + T_{cn-1} = T_{r0} + n T_{c0}

5. SUMADOR/RESTADOR EN C2 (I)

Operaciones en C2 (recordemos las diapositivas 57 y 58)

1. Cambio de signo.

Consiste en calcular el valor complementado del número.

$-A = C2(A)$

$-(-A) = C2(C2(A)) = C2(2^n - A) = 2^n - (2^n - A) = A$

2. Suma de 2 números negativos.

$(-A)+(-B)= (2^n-A) + (2^n-B) = 2^n-(A+B) + 2^n = C2(A+B) + 2^n$

Se debe ignorar 2^n. El acarreo de salida se ignora.

Ejemplo: n = 8, A = -3, B = -9

$A= 2^8-3= 11111101$; $B = 2^8-9= 11110111$

$A+B = -12= 2^8-12= 11110100$

```
  1 1 1 1 1 1 1
  1 1 1 1 1 0 1
  1 1 1 1 0 1 1 +
1 1 1 1 1 0 1 0 0
```


5. SUMADOR/RESTADOR EN C2 (II)

3. Resta (suma algebraica) de 2 números A y B

1. Considerando $|A| < |B|$.

$A-B= A+(-B)= A+(2^n-B) = 2^n-(B-A) = C2(B-A)$

Resultado a la salida es correcto.

Ejemplo: n= 8, A= 3, B= 9

$A= 00000011$; $-B= 2^8-9= 11110111$

$A-B= -6= 2^8-6= 11111010$

```
    1 1 1
0 0 0 0 0 0 1 1
1 1 1 1 0 1 1 1 +
1 1 1 1 1 0 1 0
```

2. Considerando $|A| \geq |B|$.

$A-B = A+(-B) = A + (2^n-B)= 2^n + (A-B)$

Se debe ignorar 2^n. El acarreo de salida se ignora.

Ejemplo: n = 8, A = 7, B = 3

$A = 00000111$; $-B = 2^8-3= 11111101$

$A-B = 4= 00000100$

```
  1 1 1 1 1 1 1
  0 0 0 0 0 1 1 1
  1 1 1 1 1 1 0 1 +
1 0 0 0 0 0 1 0 0
```

5. SUMADOR/RESTADOR EN C2 (III)

Para poder realizar sumas y restas aritméticas de n bits, podemos utilizar un sistema formado por n sumadores completos, al que se le ha añadido unas puertas lógicas XOR y una señal externa $\overline{S}/R$ que permite discriminar entre las dos operaciones.

$$s = XOR = \overline{b_i}\ \overline{S}/R + \overline{\overline{S}/R}\ b_i$$

si $\overline{S}/R = 0 \rightarrow sal = b_i$

si $\overline{S}/R = 1 \rightarrow sal = \overline{b_i}$

b_i	$\overline{S}/R$	s
0	0	0
0	1	1
1	0	1
1	1	0

5. SUMADOR/RESTADOR EN C2 (IV)

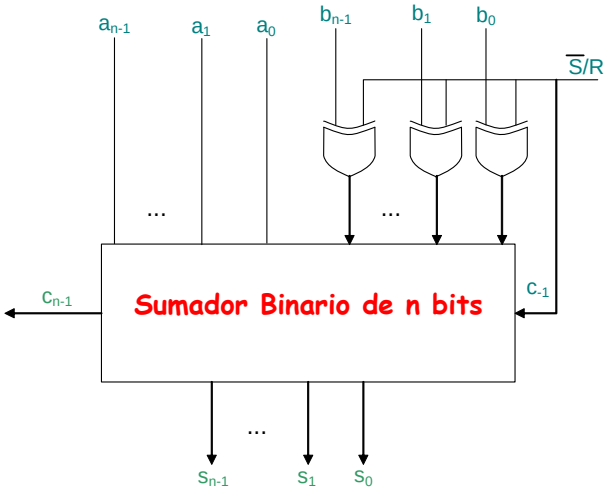
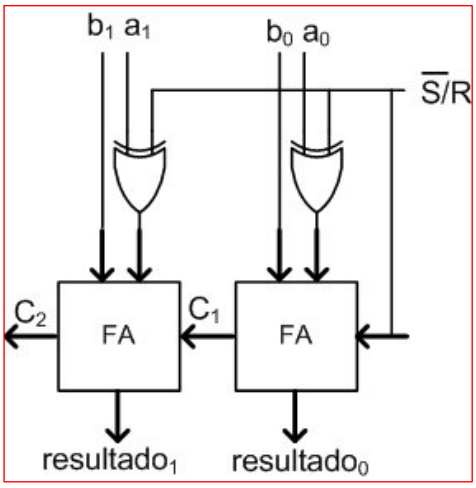
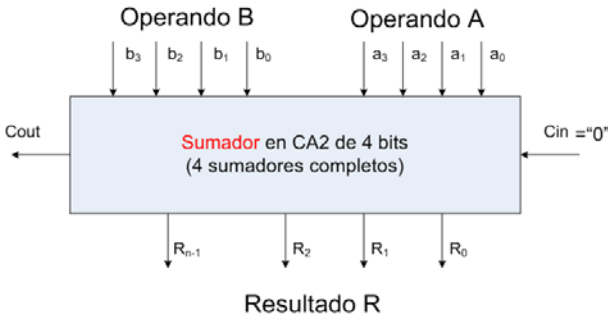


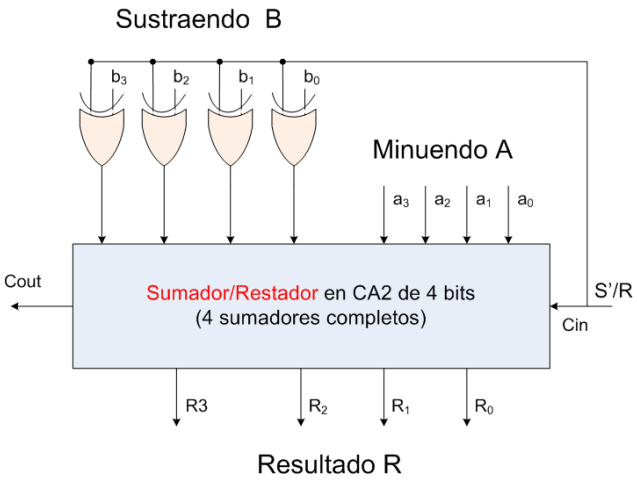
Diagrama de bloques para n bits

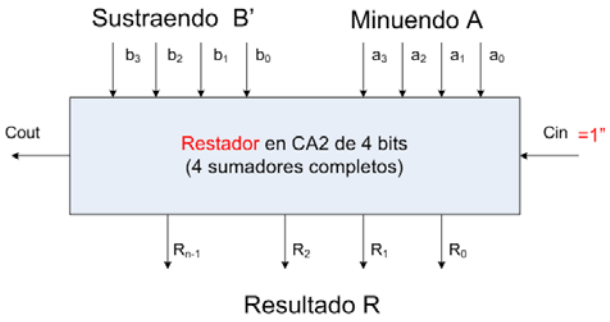


Estructura con sumadores completos para 2 bits

5. SUMADOR/RESTADOR EN C2 de 2 bits (V)







S/R	b_i	XOR	Acción
0	0	0	Pasa b_i
0	1	1	Pasa b_i
1	0	1	Complementa b_i
1	1	0	Complementa b_i

5. SUMADOR/RESTADOR EN (C2) (VI)

Condiciones de desbordamiento:

Se puede producir desbordamiento si y solo si se suman 2 cantidades positivas o 2 cantidades negativas.

- Si se suman dos cantidades **A** y **B** de n bits y con distinto signo, el resultado en valor absoluto (binario puro) será de:
  - $||A|- |B||$
  - Como $|A| < 2^n$ y $|B| < 2^n$ entonces el resultado $||A|- |B|| < 2^n$

(recordemos que 2^n es 100……0 (n); por tanto tenemos una plabra de n bits)

5. SUMADOR/RESTADOR EN (C2) (VII)

Suma de 2 cantidades positivas→ $a_{n-1}= b_{n-1}= 0$

Existirá desbordamiento cuando $c_{n-2} = 1$ (acarreo en el bit anterior al bit de signo).

El resultado será un número negativo puesto que la suma tendrá el bit $s_{n-1}= 1$.

$$a_{n-1} + b_{n-1} + c_{n-2} = 0+0+1 =1$$

Suma de 2 cantidades negativas→ $a_{n-1} = b_{n-1} = 1$

Existirá desbordamiento cuando $c_{n-2}= 0$ (no existe acarreo en el anterior bit al bit de signo).

El resultado será un número positivo puesto que la suma tendrá el bit $s_{n-1}= 0$.

$$a_{n-1} + b_{n-1} + c_{n-2} = 1+1+0 =0$$

5. SUMADOR/RESTADOR EN (C2) (VIII)

Condición de desbordamiento:

$$DE= \overline{a_{n-1}} \cdot \overline{b_{n-1}} \cdot c_{n-2} + a_{n-1} \cdot b_{n-1} \cdot \overline{c_{n-2}}$$

Teniendo en cuenta que:

$$c_{n-1}= a_{n-1} \cdot b_{n-1} + a_{n-1} \cdot c_{n-2} + b_{n-1} \cdot c_{n-2}$$

La expresión inicial del desbordamiento queda:

$$DE= c_{n-1} \oplus c_{n-2}$$

5. SUMADOR/RESTADOR EN (C2) (IX)

Ejemplos :

$$DE = \overline{a_{n-1}} \cdot \overline{b_{n-1}} \cdot c_{n-2} + a_{n-1} \cdot b_{n-1} \cdot \overline{c_{n-2}}$$

2 positivos : +13 y +10

$c_{n-2} = 1 \quad a_{n-1} = 0 \quad b_{n-1} = 0$

DE = 1

	1	0	0	0	
	0	1	1	0	1
	0	1	0	1	0 +
	1	0	1	1	1

2 negativos: -13 y -10

$c_{n-2} = 0 \quad a_{n-1} = 1 \; b_{n-1} = 1$

DE = 1

	0	1	1	0	
	1	0	0	1	1
	1	0	1	1	0 +
1	0	1	0	0	1